Kafka Messaging - Developer Guide

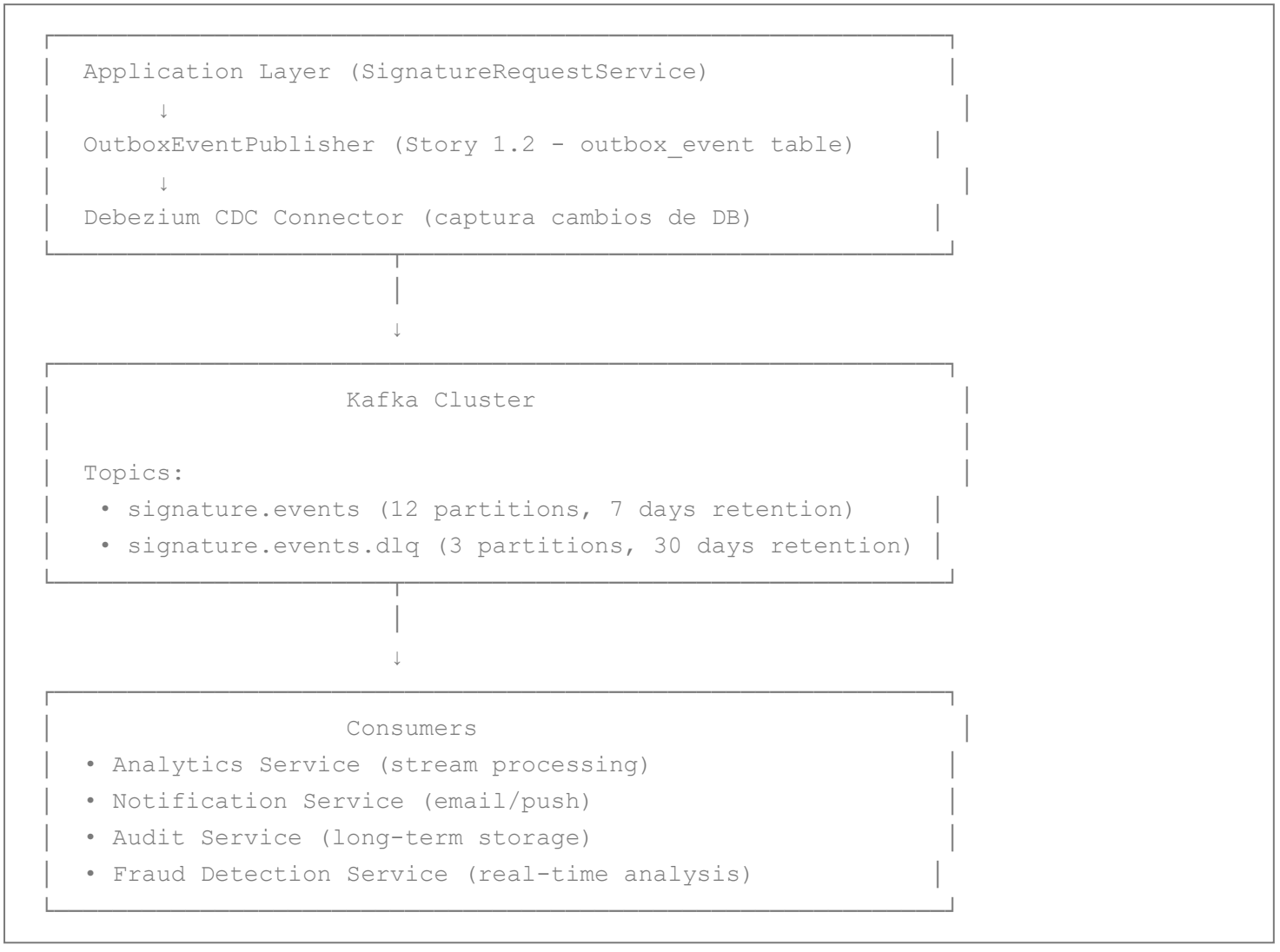
Version: 1.0
Date: 2025-11-26
Status: Ready for Development
Story: 1.3 - Kafka Infrastructure & Schema Registry

---

1. Overview

El **Signature Router** utiliza **Apache Kafka** como backbone para event-driven architecture. Todos los eventos de dominio se publican a Kafka utilizando **Avro serialization** con **Schema Registry** para garantizar validación de esquemas y backward compatibility.

Arquitectura de Eventos



---

2. Quick Start

2.1 Local Development Setup

```
# 1. Start Kafka cluster (Zookeeper + Kafka + Schema Registry)
docker-compose up -d zookeeper kafka schema-registry

# 2. Verify services are running
docker ps | grep signature-router

# 3. Check Kafka broker health
docker exec signature-router-kafka kafka-broker-api-versions --bootstrap-server
localhost:9092

# 4. Check Schema Registry health
curl http://localhost:8081/
```

2.2 Run Application

```
# Compile (generates Avro classes from .avsc schemas)
./mvnw clean compile

# Run with local profile
./mvnw spring-boot:run -Dspring-boot.run.profiles=local

# Verify Kafka health check
curl http://localhost:8080/actuator/health/kafka
```

---

3. Event Publishing

3.1 Event Types

Event Type	Trigger	Payload Fields
`SIGNATURE_REQUEST_CREATED`	SignatureRequest created	customerId, requestStatus
`CHALLENGE_SENT`	Challenge delivered to provider	channel, provider
`CHALLENGE_COMPLETED`	User completed challenge	channel, durationMs
`CHALLENGE_FAILED`	Provider rejected challenge	errorCode, errorMessage

Event Type	Trigger	Payload Fields
SIGNATURE_COMPLETED	User completed signature	durationMs, attemptCount
SIGNATURE_FAILED	Signature process failed	errorCode, errorMessage
FALLBACK_TRIGGERED	Fallback channel activated	channel, provider
PROVIDER_DEGRADED	Provider circuit breaker opened	provider, errorCode

3.2 Publishing Events (Example)

```
@Service
public class SignatureEventPublisher {

    private final KafkaTemplate<String, GenericRecord> kafkaTemplate;

    @Autowired
    public SignatureEventPublisher(KafkaTemplate<String, GenericRecord>
kafkaTemplate) {
        this.kafkaTemplate = kafkaTemplate;
    }

    public void publishSignatureRequestCreated(SignatureRequest request) {
        // Build event payload
        EventPayload payload = EventPayload.newBuilder()
                .setCustomerId(request.getPseudonymizedCustomerId())
                .setRequestStatus(request.getStatus().name())
                .setChannel(null) // Not applicable for REQUEST_CREATED
                .setProvider(null)
                .setErrorCode(null)
                .setErrorMessage(null)
                .setAttemptCount(0)
                .setDurationMs(null)
                .build();

        // Build SignatureEvent
        SignatureEvent event = SignatureEvent.newBuilder()
                .setEventId(UUID.randomUUID().toString()) // UUIDv7 in production
                .setEventType(EventType.SIGNATURE_REQUEST_CREATED)
                .setAggregateId(request.getId().toString())
                .setAggregateType("SignatureRequest")
                .setTimestamp(Instant.now().toEpochMilli())
                .setTraceId(MDC.get("traceId")) // From OpenTelemetry
```

```
                .setPayload(payload)
                .build();

        // Send to Kafka (partition key = aggregateId for ordering)
        kafkaTemplate.send("signature.events", request.getId().toString(), event);
    }
}
```

---

4. Schema Management

4.1 Avro Schema Location

```
src/main/resources/kafka/schemas/signature-event.avsc
```

4.2 Schema Evolution Guidelines

✅ Allowed Changes (Backward Compatible)

1. **Add optional field** (with default value):

```
{
  "name": "newField",
  "type": ["null", "string"],
  "default": null,
  "doc": "New optional field"
}
```

2. **Add new enum value** (at the end):

```
{
  "type": "enum",
  "name": "EventType",
  "symbols": [
    "SIGNATURE_REQUEST_CREATED",
    "CHALLENGE_SENT",
    "NEW_EVENT_TYPE"  // OK: add at end
  ]
}
```

❌ Forbidden Changes (Breaking Compatibility)

1. ❌ Remove existing field
2. ❌ Change field type (e.g., `string` → `int`)
3. ❌ Rename field
4. ❌ Remove enum value

5. ❌ Add required field (without default)

4.3 Updating Schema

```
# 1. Edit schema file
vi src/main/resources/kafka/schemas/signature-event.avsc

# 2. Recompile (Maven Avro Plugin regenerates Java classes)
./mvnw clean compile

# 3. Verify generated classes
ls target/generated-sources/avro/com/bank/signature/event/

# 4. Run tests
./mvnw test

# 5. Schema Registry validates compatibility on first publish
```

4.4 Manual Schema Registration (Optional)

```
# Register schema manually in Schema Registry
curl -X POST http://localhost:8081/subjects/signature.events-value/versions \
  -H "Content-Type: application/vnd.schemaregistry.v1+json" \
  --data @- <<EOF
{
  "schema": "$(cat src/main/resources/kafka/schemas/signature-event.avsc | jq -c .
| jq -R .)"
}
EOF

# List schemas
curl http://localhost:8081/subjects

# Get latest schema version
curl http://localhost:8081/subjects/signature.events-value/versions/latest
```

---

5. Kafka Administration

5.1 List Topics

```
docker exec signature-router-kafka kafka-topics \
  --bootstrap-server localhost:9092 \
  --list
```

5.2 Describe Topic (signature.events)

```
docker exec signature-router-kafka kafka-topics \
  --bootstrap-server localhost:9092 \
  --describe \
  --topic signature.events
```

5.3 Consume Messages (Manual Testing)

```
# Consume from beginning (Avro messages - binary output)
docker exec signature-router-kafka kafka-console-consumer \
  --bootstrap-server localhost:9092 \
  --topic signature.events \
  --from-beginning

# Consume with Avro deserializer (requires Schema Registry)
docker exec signature-router-kafka kafka-avro-console-consumer \
  --bootstrap-server localhost:9092 \
  --topic signature.events \
  --property schema.registry.url=http://schema-registry:8081 \
  --from-beginning
```

5.4 Check Consumer Lag

```
docker exec signature-router-kafka kafka-consumer-groups \
  --bootstrap-server localhost:9092 \
  --describe \
  --group signature-router-consumer-group
```

---

6. Troubleshooting

6.1 Kafka Not Starting

Síntoma: `docker-compose up` falla con error en Kafka

Solución:

```
# 1. Verify Zookeeper is running first
docker logs signature-router-zookeeper

# 2. Check Kafka logs
docker logs signature-router-kafka

# 3. Restart services in order
docker-compose restart zookeeper
sleep 10
docker-compose restart kafka
sleep 10
docker-compose restart schema-registry
```

6.2 Schema Registry Connection Refused

Síntoma: `Connection refused: http://localhost:8081`

Solución:

```
# 1. Verify Schema Registry is running
docker ps | grep schema-registry

# 2. Check health endpoint
curl http://localhost:8081/

# 3. Check Schema Registry logs
docker logs signature-router-schema-registry

# 4. Verify Kafka is reachable from Schema Registry
docker exec signature-router-schema-registry ping kafka
```

6.3 Avro Classes Not Generated

Síntoma: `Cannot resolve symbol 'SignatureEvent'`

Solución:

```
# 1. Clean and recompile
./mvnw clean compile

# 2. Verify Avro Maven Plugin execution
./mvnw clean compile -X | grep avro-maven-plugin

# 3. Check generated sources
ls -la target/generated-sources/avro/com/bank/signature/event/

# 4. Refresh IDE (IntelliJ: Maven -> Reload Project)
```

6.4 Schema Incompatibility Error

Síntoma: `Schema being registered is incompatible with an earlier schema`

Solución:

```
# 1. Check current schema in Schema Registry
curl http://localhost:8081/subjects/signature.events-value/versions/latest

# 2. Compare with local schema
cat src/main/resources/kafka/schemas/signature-event.avsc

# 3. Fix compatibility issues (add default values, don't remove fields)

# 4. Delete subject (DEV ONLY - DANGEROUS IN PROD)
curl -X DELETE http://localhost:8081/subjects/signature.events-value
```

6.5 Health Check Failing

Síntoma: `/actuator/health/kafka` returns `DOWN`

Solución:

```
# 1. Verify Kafka broker is reachable
telnet localhost 9092

# 2. Check application-local.yml configuration
grep bootstrap-servers src/main/resources/application-local.yml

# 3. Verify KafkaTemplate bean is created
curl http://localhost:8080/actuator/beans | jq '.contexts."signature-router".beans
| with_entries(select(.key | contains("kafka")))'

# 4. Check application logs
tail -f logs/signature-router.log | grep -i kafka
```

---

7. Testing

7.1 Run Integration Tests

```
# Run all tests (includes Embedded Kafka tests)
./mvnw test

# Run only Kafka tests
./mvnw test -Dtest=KafkaInfrastructureIntegrationTest
```

7.2 Manual Event Publishing (cURL)

```
# Note: Direct REST API for event publishing is not exposed.
# Events are published internally via application logic.
# For manual testing, use Spring Boot application and trigger domain actions.
```

---

8. Production Considerations

8.1 Topic Configuration

Environment	Partitions	Replication Factor	Retention
Dev	12	1	7 days
UAT	12	2	7 days
Prod	12	3	7 days

8.2 Producer Configuration

Property	Dev	Prod	Rationale
`acks`	all	all	Banking-grade durability
`enable.idempotence`	true	true	Exactly-once semantics
`compression.type`	snappy	snappy	Network efficiency
`max.in.flight.requests.per.connection`	5	5	Balance throughput vs. ordering
`retries`	MAX_VALUE	MAX_VALUE	Infinite retries with backoff

8.3 Monitoring (Future - Story 2.x)

- **Prometheus Metrics:** Kafka producer/consumer metrics
- **Grafana Dashboards:** Kafka lag, throughput, error rate
- **Alerts:** Consumer lag > threshold, Schema Registry errors

---

9. References

- **Confluent Platform Documentation:** https://docs.confluent.io/
- **Apache Avro Specification:** https://avro.apache.org/docs/current/spec.html
- **Spring Kafka Documentation:** https://docs.spring.io/spring-kafka/reference/html/
- **Architecture:** `docs/architecture/04-event-catalog.md`
- **Tech Spec Epic 1:** `docs/sprint-artifacts/tech-spec-epic-1.md`

---

10. Quick Commands Reference

```
# Start Kafka infrastructure
docker-compose up -d zookeeper kafka schema-registry

# Stop Kafka infrastructure
docker-compose stop zookeeper kafka schema-registry

# View Kafka logs
docker logs -f signature-router-kafka

# List topics
docker exec signature-router-kafka kafka-topics --bootstrap-server localhost:9092
--list

# Describe topic
docker exec signature-router-kafka kafka-topics --bootstrap-server localhost:9092
--describe --topic signature.events

# Consume messages
docker exec signature-router-kafka kafka-console-consumer --bootstrap-server
localhost:9092 --topic signature.events --from-beginning

# Check Schema Registry schemas
curl http://localhost:8081/subjects

# Get latest schema
curl http://localhost:8081/subjects/signature.events-value/versions/latest | jq .

# Kafka health check
curl http://localhost:8080/actuator/health/kafka | jq .

# Compile (generate Avro classes)
./mvnw clean compile

# Run tests
```

```
./mvnw test -Dtest=KafkaInfrastructureIntegrationTest
```

Last Updated: 2025-11-26 (Story 1.3)
Maintainer: Signature Router Team